多媒體技術與應用
Spring 2021

Instructor：Yen-Lin Chen(陳彥霖), Ph.D.
Professor
Dept. Computer Science and Information Engineering
National Taipei University of Technology

Lecture 10

RNN與LSTM的發展與應用

RNN循環神經網路 (Recurrent neural network)

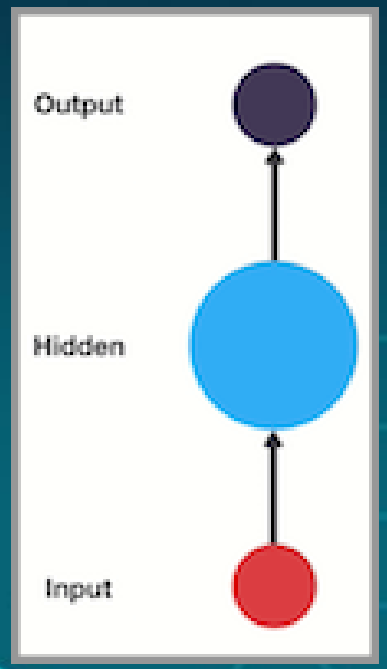
RNN簡介

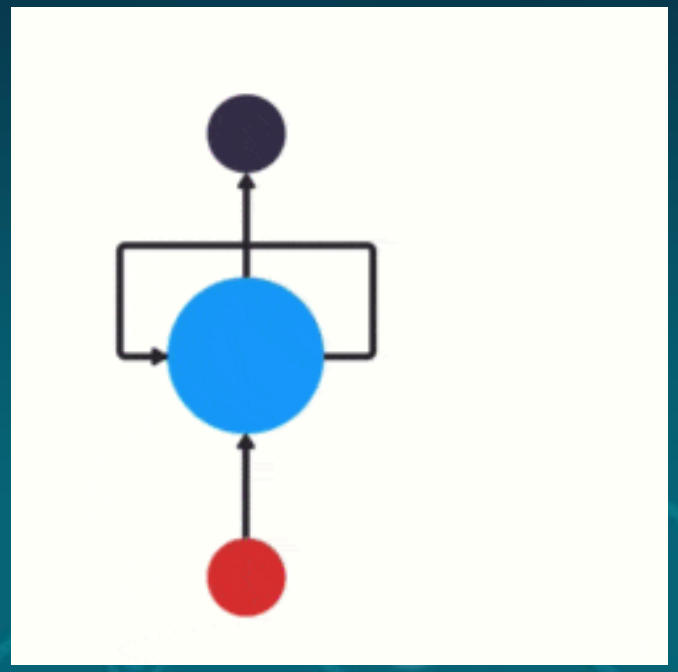
- **循環神經網路**（Recurrent neural network：**RNN**）是神經網路的一種。可以描述動態時間行為，因為和前饋神經網路(例如我們先前所教過的CNN) 僅能接受較特定結構的輸入不同，RNN將狀態在自身網路中循環傳遞，因此可以接受更廣泛的時間序列結構輸入。手寫識別是最早成功利用RNN的研究結果。
- 循環神經網路單純的RNN因為無法處理梯度爆炸或梯度消失問題，難以捕捉長期時間關聯，因此需產生數種不同的衍伸模型。

Ref: https://codingnote.cc/zh-tw/p/8153/

RNN簡介

- 傳統的神經網路(ex. CNN)與循環神經網路RNN的結構差異。
- 左圖為傳統神經網路，分為輸入層、隱藏層、輸出層，右圖為循環神經網路，可以見到會將每一次的輸出結果，帶入到下一次的隱藏層中一起訓練。





Ref: https://codingnote.cc/zh-tw/p/8153/

RNN發展歷史

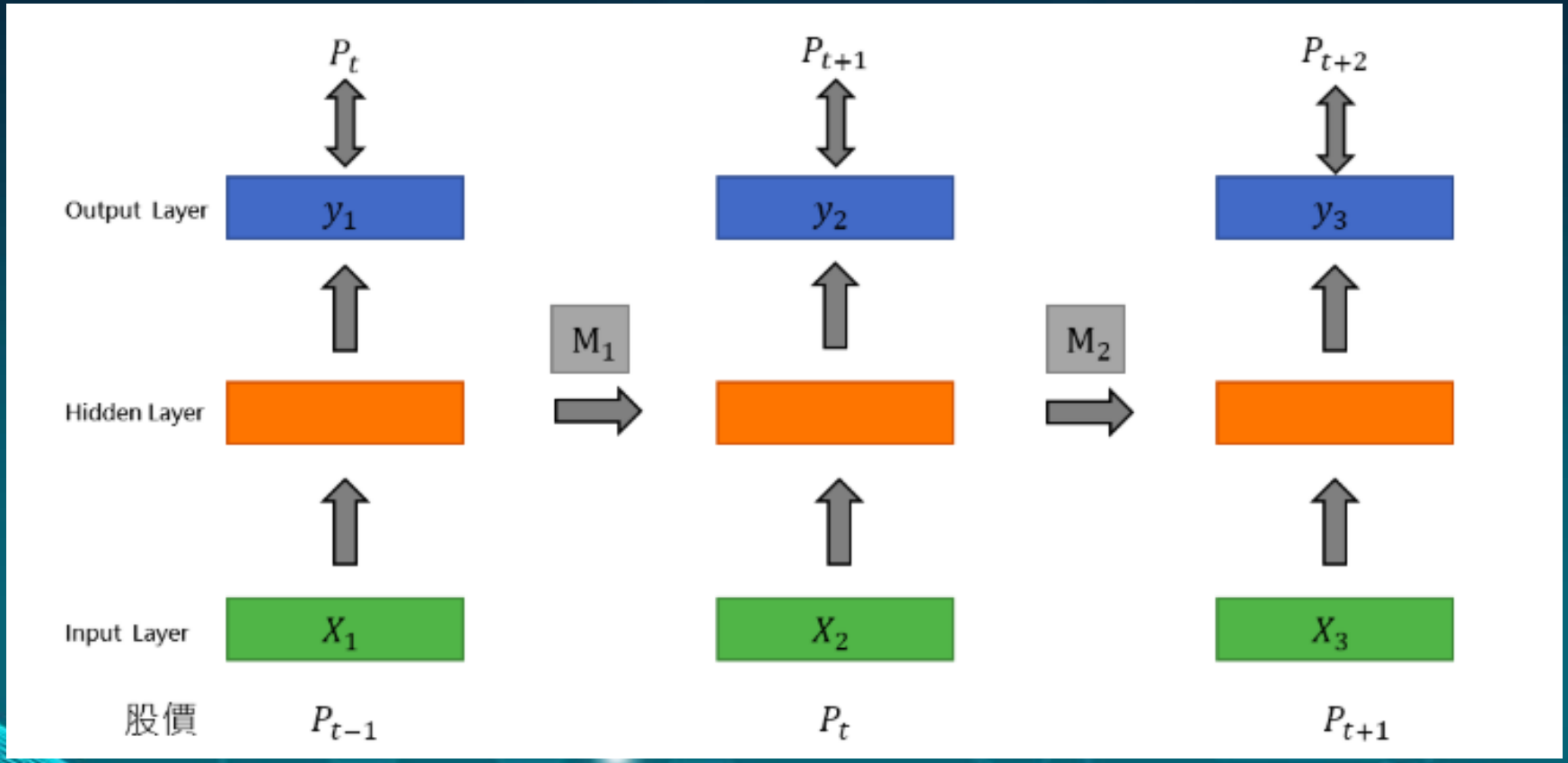
- 1982年，美國加州理工學院物理學家John Hopfield發明了一種單層反饋神經網絡Hopfield Network，用來解決組合最佳化問題。這是最早的RNN的雛形。
- 1986年，另一位機器學習的泰斗Michael I. Jordan定義了Recurrent的概念，提出Jordan Network。
- 1990年，美國認知科學家Jeffrey L. Elman對Jordan Network進行了簡化,並採用BP算法進行訓練。
- 1997年，瑞士人工智能研究所的主任Jurgen Schmidhuber提出長短期記憶（LSTM），LSTM使用**門控單元及記憶機制**大大緩解了早期RNN訓練的問題。

RNN發展歷史

- 1997年，Mike Schuster 提出雙向RNN模型（Bidirectional RNN）。
- 2010年，Tomas Mikolov對Bengio等人提出的feedforward Neural network language model (NNLM)進行了改進，提出了基於RNN的語言模型（RNN LM），並將其用在語音識別任務中，大幅提升了識別精度，在此基礎上Tomas Mikolov於2013年提出了大名鼎鼎的word2vec。
- 2014年Bengio團隊與Google幾乎同時提出了seq2seq架構，將RNN用於機器翻譯。

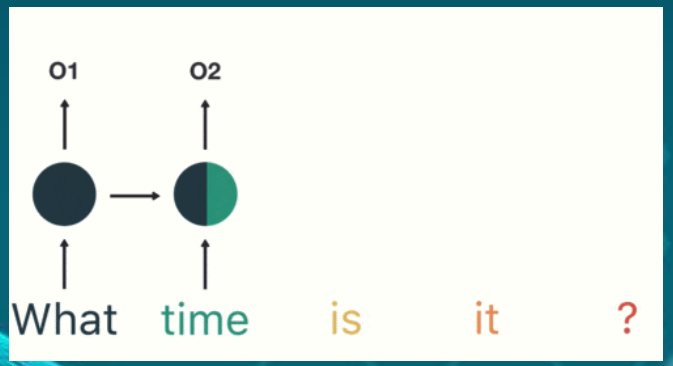
RNN的基本運作方式

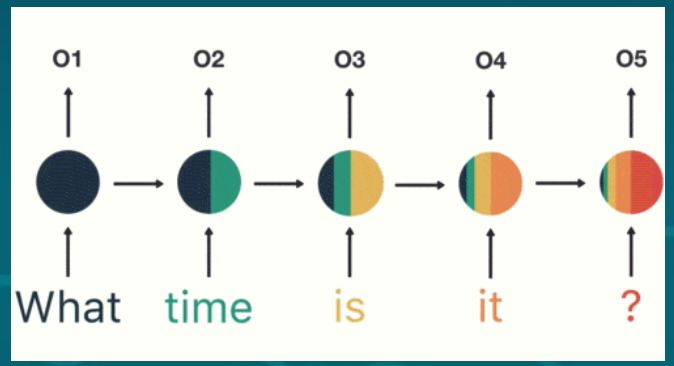
- 將前一次隱藏層的output存在Memory中，並在下一次訓練時納入考慮。



RNN的基本運作方式(以語意為例子)

- 分詞>輸入模型>以前一次模型訓練結果影響下一次的模型>最終輸出(判斷整句的意圖)

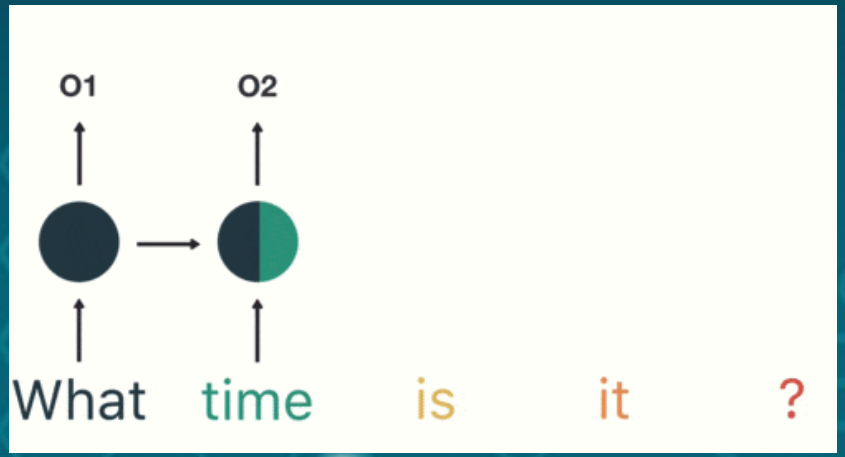




圖源: https://easyai.tech/ai-definition/rnn/?fbclid=IwAR2Wt9iF0vSbshRBLWr3gt1PnJ1G7knFMPNd8PJhJJ2swQZkwWLxkqXCRMw#jiazhi

RNN的主要缺點

- RNN 在長期記憶的表現並不如預期。因為其結構特性，使得RNN呈現「短期記憶影響較大，長期記憶影響較小(如範例圖中，較久遠的黑色記憶影響較小，而新加入的記憶影響較大)」，並且隨著訓練時間加長與網路層數增加等等原因，會產生梯度爆炸或梯度消失的問題(神經網路中，前後層之間變化速率不同，使得整體快速趨近於沒有變化或是變化過大無法穩定在最佳點的現象)，為了解決這類問題，隨之誕生的就是LSTM。



圖源: https://easyai.tech/ai-definition/rnn/?fbclid=IwAR2Wt9iF0vSbshRBLWr3gt1PnJ1G7knFMPNd8PJhJJ2swQZkwWLxkqXCRMw#jiazhi

LSTM長短期記憶神經網路 (Long Short-Term Memory)

LSTM簡介

- 長短期記憶神經網路（Long Short-Term Memory，LSTM）是一種特殊的循環神經網路(Recurrent neural network，RNN)。原始的RNN在訓練中，隨著訓練時間的加長以及網路層數的增多，很容易出現梯度爆炸或者梯度消失的問題，導致無法處理較長序列數據，從而無法獲取長距離數據的資訊

- LSTM應用的領域包括：文句生成、機器翻譯、語音識別、生成影像描述和影片標記等。

Ref: https://codingnote.cc/zh-tw/p/8153/

LSTM發展歷史

- 1997年，Sepp Hochreiter 和 Jürgen Schmidhuber提出了長短期記憶神經網路(LSTM)，有效解決了RNN難以解決的人為延長時間任務的問題，並解決了RNN容易出現梯度消失的問題。

- 1999年，Felix A. Gers等人發現Sepp Hochreiter提出的LSTM在處理連續輸入數據時，如果沒有重置網路內部的狀態，最終會導致網路崩潰。因此，他們在Sepp Hochreiter的文獻基礎上引入了**遺忘門**機制，使得LSTM能夠重置自己的狀態。

LSTM發展歷史

- 2000年，Felix A. Gers和Jiirgen Schmidhuber發現，通過在LSTM內部狀態單元內添加窺視孔（Peephole）連接，可以增強網路對輸入序列之間細微特徵的區分能力。

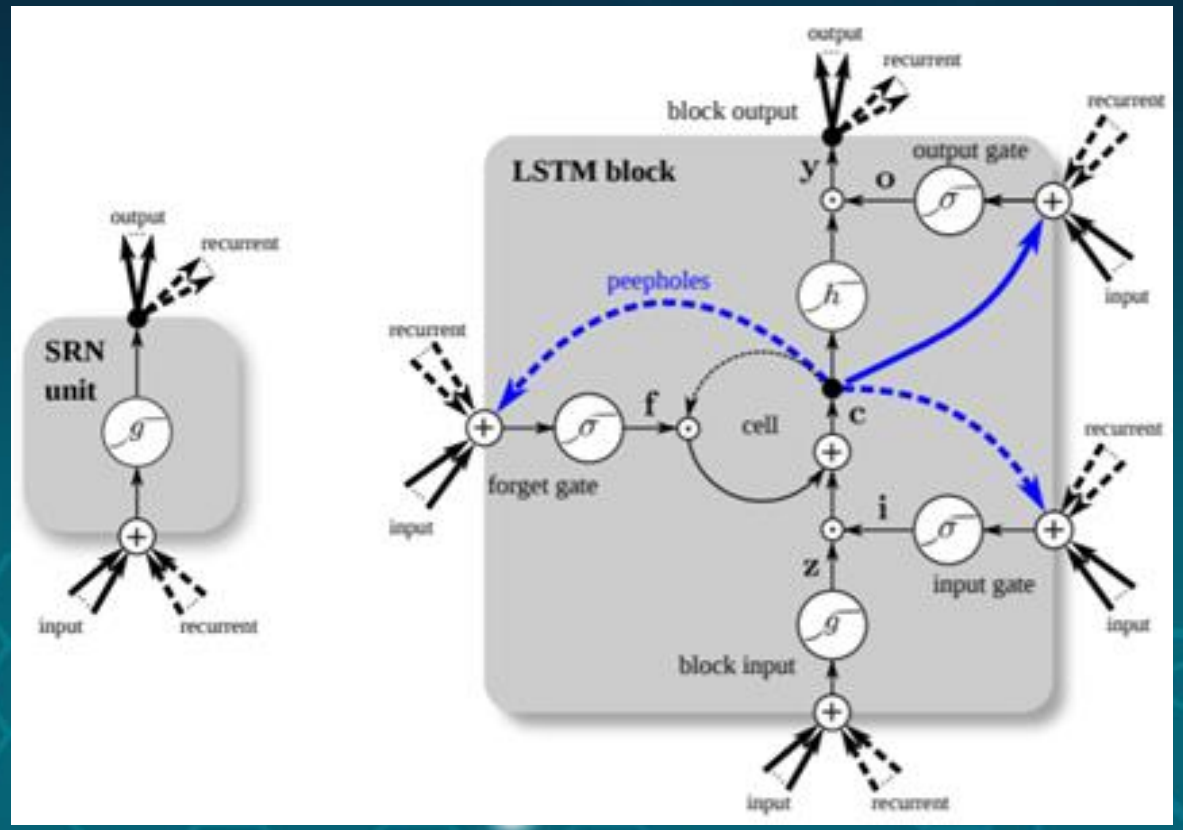
- 2005年，Alex Graves和Jürgen Schmidhuber在過去文獻的基礎上提出了一種雙向長短期記憶神經網路（BLSTM），也稱為vanilla LSTM，是當前應用最廣泛的一種LSTM模型。

LSTM發展歷史

- 2016年，Klaus Greff 等人回顧了LSTM的發展歷程，並比較分析了八種LSTM變體在語音識別、手寫識別和弦音樂建模方面的能力，實驗結果證明了**遺忘門**和**輸出啟動**功能是LSTM的關鍵組成部分。在這八種變體中，vanilla LSTM的綜合表現能力最佳。另外，還探索了LSTM相關超參數(Hyperparameters，人為設定的參數)的設定影響，實驗結果表明學習速率(learning rate)是最關鍵的超參數，其次是網路規模（網路層數和隱藏層單元數），而動量梯度 (Gradient descent with Momentum動量梯度下降法等等方法)等設置對最終結果影響不大。

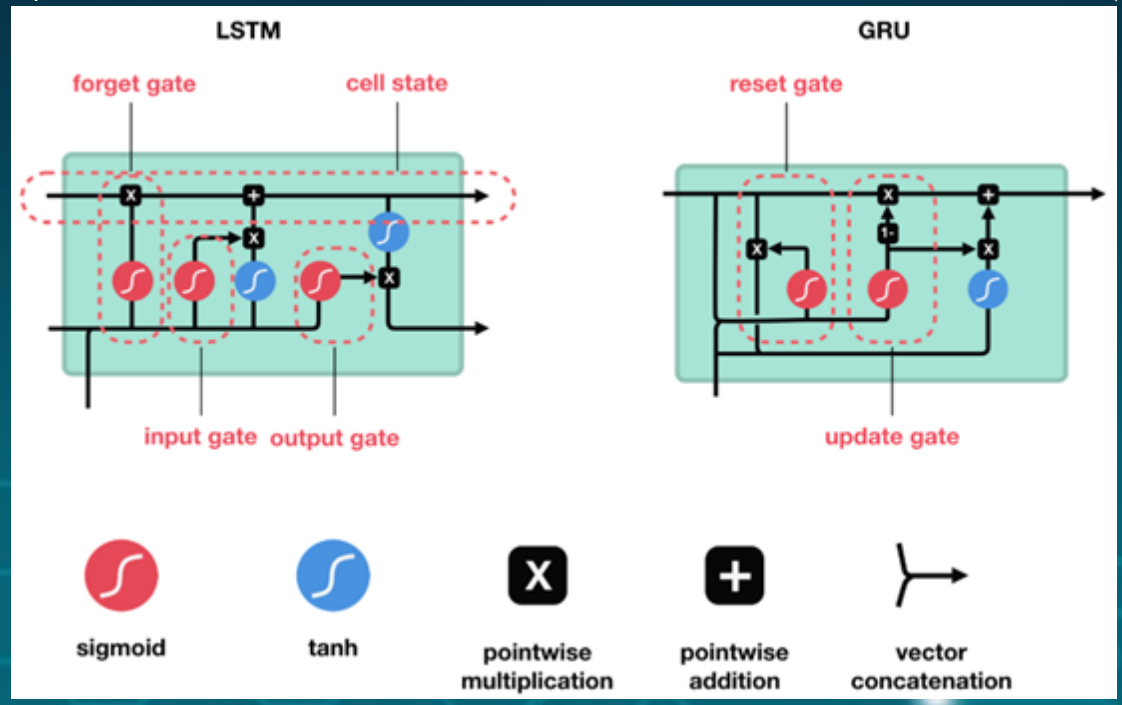
LSTM發展歷史

- 下圖展示了Simple RNN（圖左）和vanilla LSTM（圖右，圖中藍色線條表示窺視孔連接）的基本單元結構圖：



LSTM發展歷史

- 在眾多LSTM變體中，2014年Kyunghyun Cho等人[6]提出的變體引起了眾多學者的關注。Kyunghyun Cho等人簡化了LSTM架構，稱為**門控遞迴單元**（GRU）。GRU擺脫了單元狀態，基本結構由**重置門**和**更新門**組成。LSTM和GRU的基本結構單元如下圖。



- （神經網路結構具體可參考：Illustrated Guide to LSTM's and GRU's: A step by step explanation）。

LSTM發展歷史

- 在GRU被提出後，Junyoung Chung等人比較了LSTM和GRU在複音音樂(多聲部音樂，作品中含有兩條以上獨立旋律，通過技術性處理，和諧地結合在一起)和語音訊號建模方面的能力，實驗結果表明GRU和LSTM表現相當。

- GRU被提出至今（2019年），也只有幾年時間，關於它的一些應用利弊到目前還未探索清楚。不過，相對於LSTM架構，GRU的的參數較少，在數據量較大的情況下，其訓練速度更快。

LSTM發展歷史

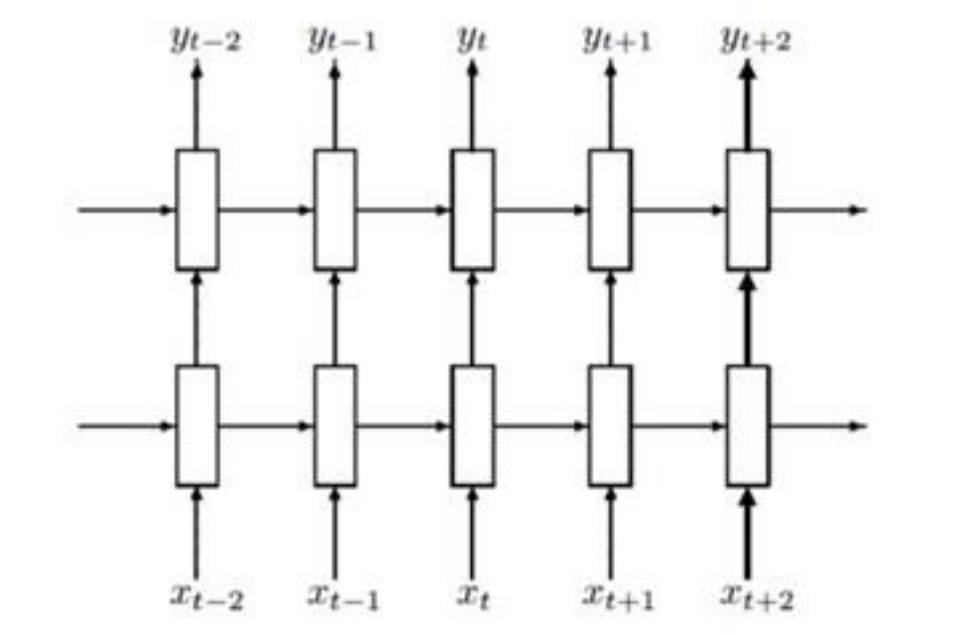
- LSTM是深度學習技術中的一員，其基本結構比較複雜，計算複雜度較高，導致較難進行較深層次的學習，例如Google翻譯也只是應用7-8層的LSTM網路結構。另外，在訓練學習過程中有可能會出現overfitting，可以通過應用dropout(Google提出的一種正則化技術)來解決overfitting問題。

- LSTM在當前應用比較的結構是雙向LSTM或者多層堆疊LSTM，這兩種結構的實現在Keras等框架中均有對應的API可以調用。

- Dropout技術在Keras等框架中均有實現，具體可參考：LSTM原理與實踐，原來如此簡單）

LSTM發展歷史

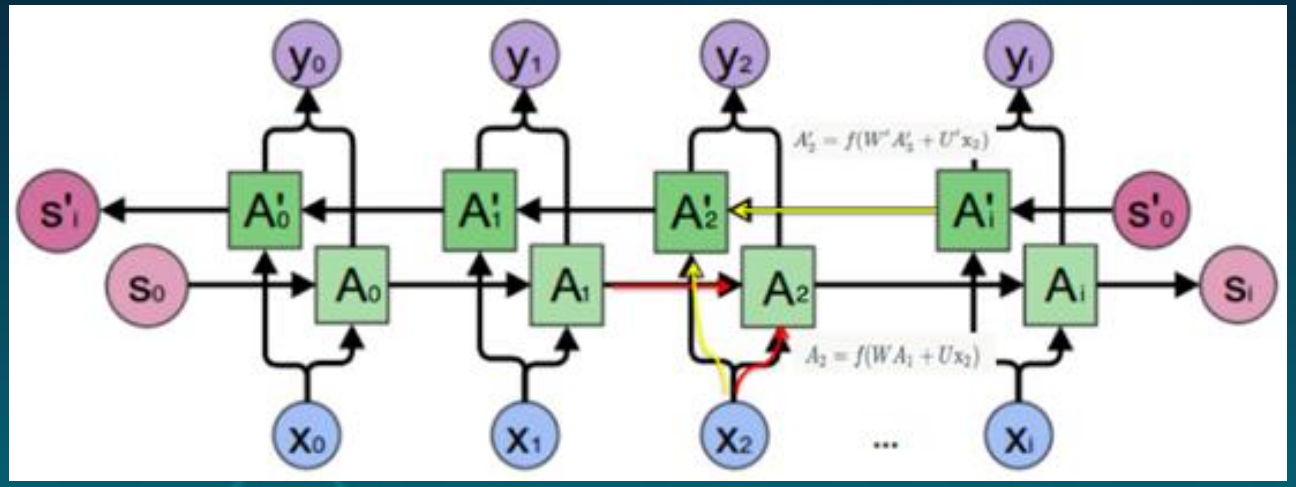
- 下圖展示一個堆疊兩層的LSTM結構圖：



- 來源：運用TensorFlow處理簡單的NLP問題

LSTM發展歷史

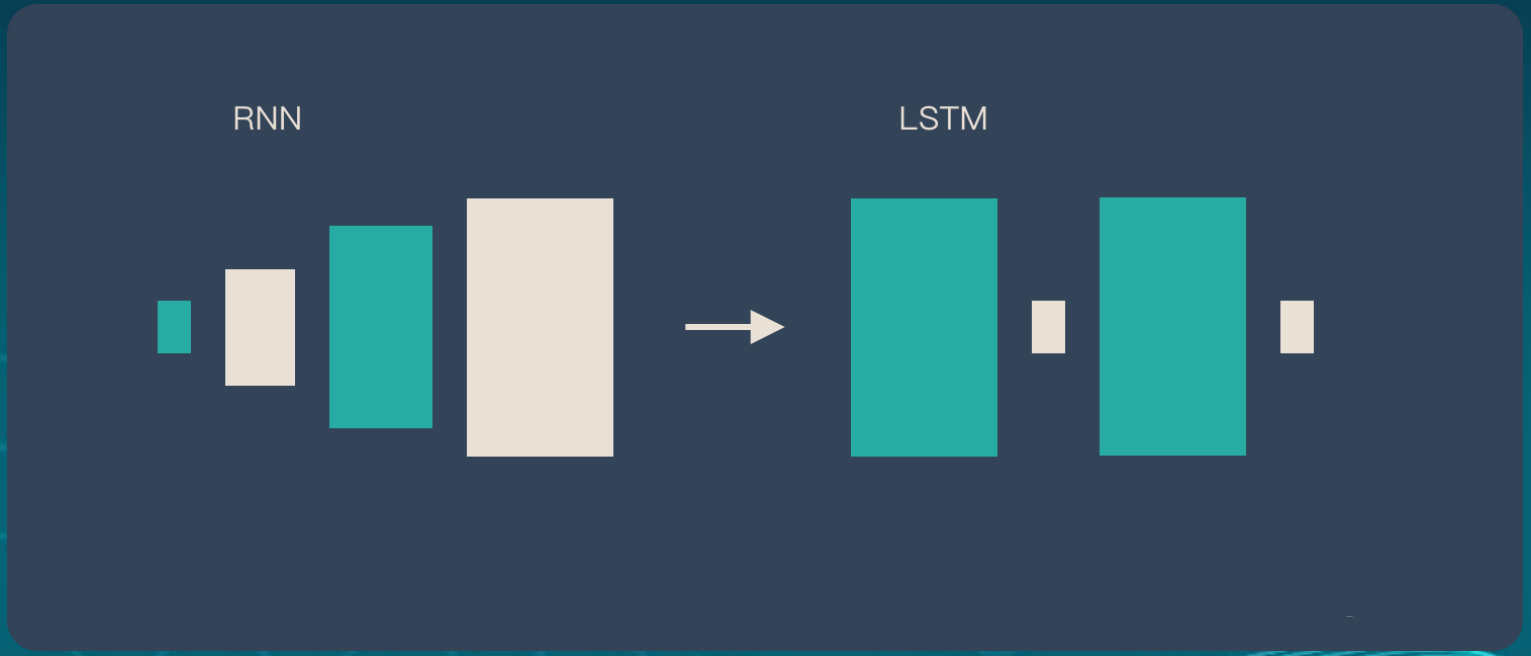
- 下圖展示了一個雙向LSTM的結構圖



- 來源：雙向LSTM

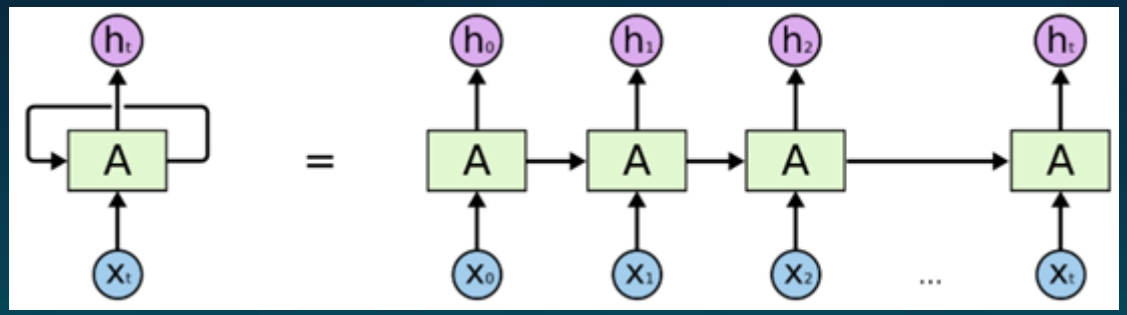
LSTM基本原理

- 本節首先講解一下RNN的基本結構，然後說明LSTM的具體原理（下面要介紹的LSTM即為vanilla LSTM）。
- LSTM比起RNN，做的行為即為「抓重點」，可以見到RNN中，純粹依據時間序列決定重要性，而LSTM則將綠色(重點訊息)的部分進行強調。



LSTM基本原理

- 原始的RNN基本結構圖如下圖所示。



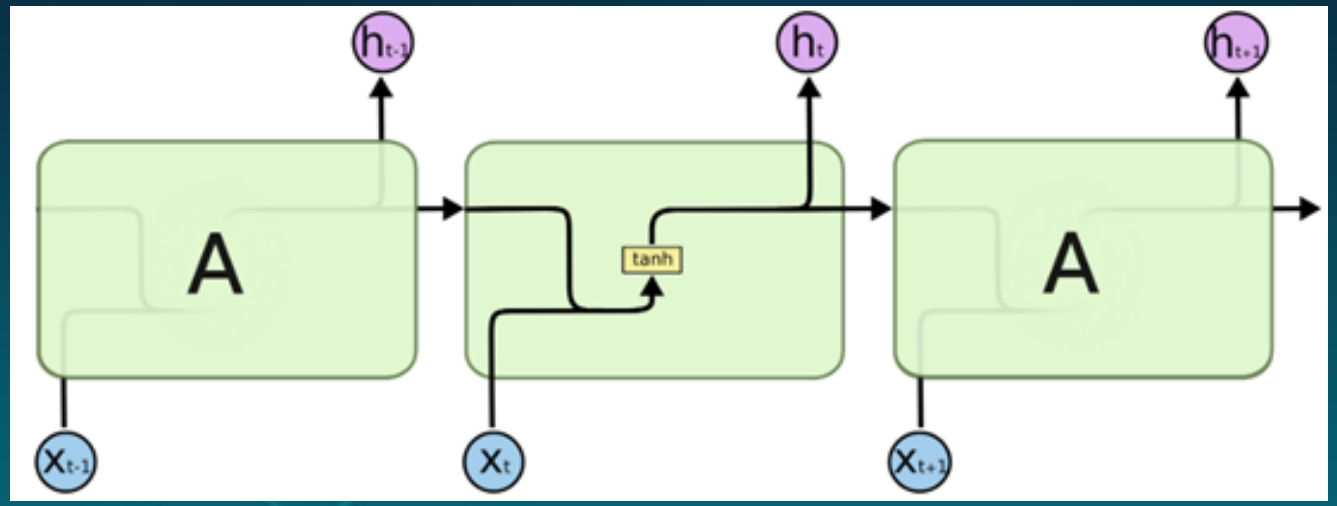
- 由上圖可知，RNN展開後由多個相同的單元連續連接。但是，RNN的實際結構卻和上圖左邊的結構所示，是一個自我不斷循環的結構。即隨著輸入數據的不斷增加，上述自我循環的結構把上一次的狀態傳遞給當前輸入，一起作為新的輸入數據進行當前輪次的訓練和學習，一直到輸入或者訓練結束，最終得到的輸出即為整體的預測結果。
- 原圖來源：Understanding LSTM Networks

LSTM基本原理

- LSTM是一種特殊的RNN，兩者的區別在於普通的RNN單個循環結構內部只有一個狀態。而LSTM的單個循環結構(又稱為cell)內部有四個狀態。相比於RNN，LSTM循環結構之間保持一個持久的單元狀態不斷傳遞下去，用於決定哪些資訊要遺忘或者繼續傳遞下去。

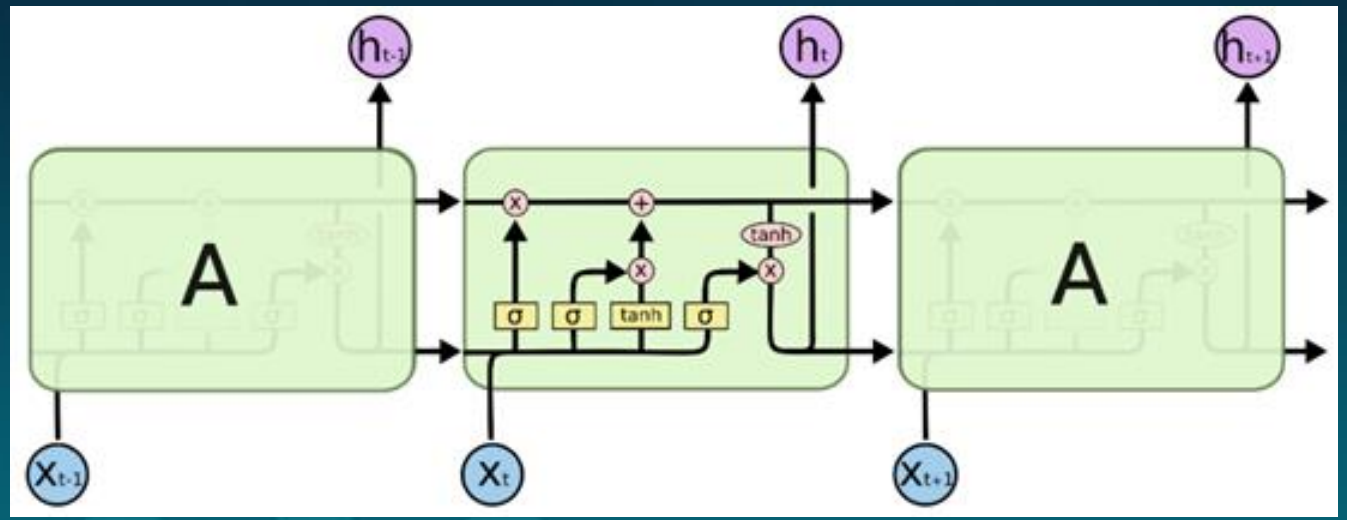
LSTM基本原理

- 包含三個連續循環結構的RNN如下圖，每個循環結構只有一個輸出：



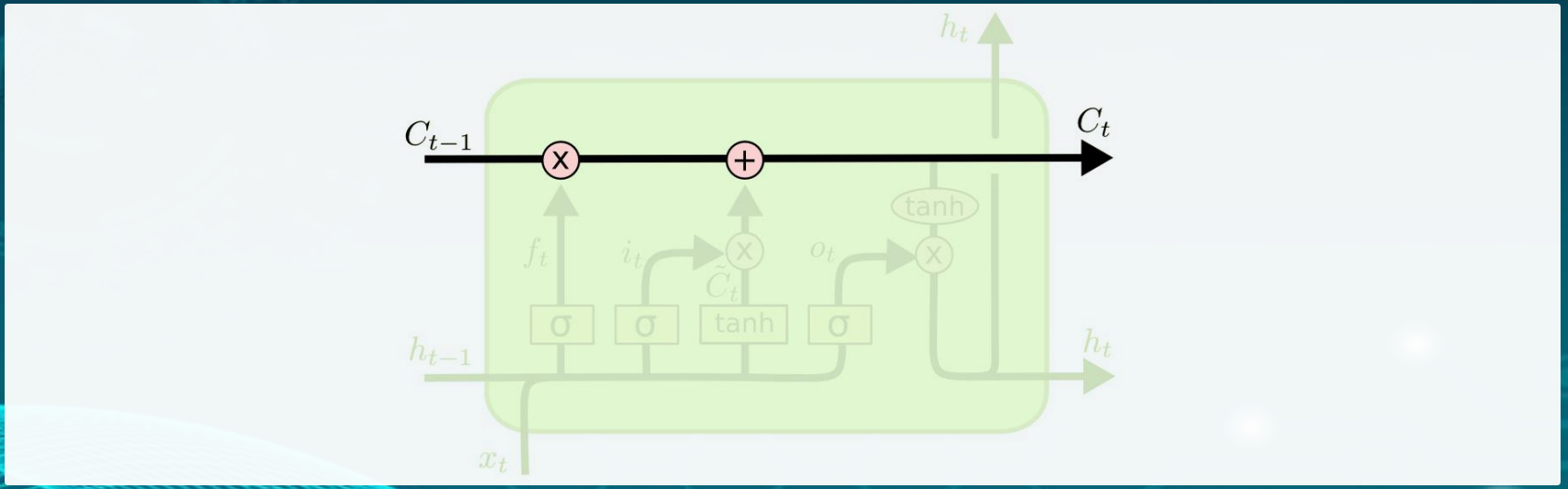
LSTM基本原理

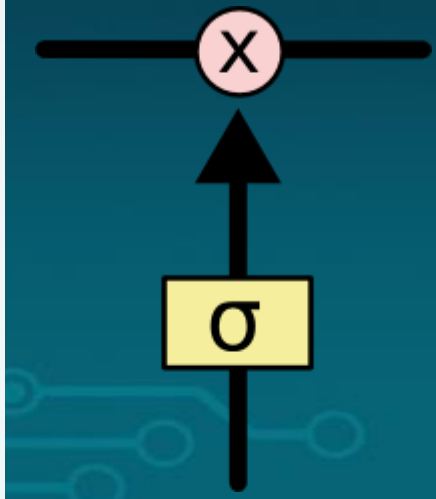
- 包含三個連續循環結構的LSTM如下圖，每個循環結構有兩個輸出，其中一個輸出即為單元狀態：



LSTM-單元狀態

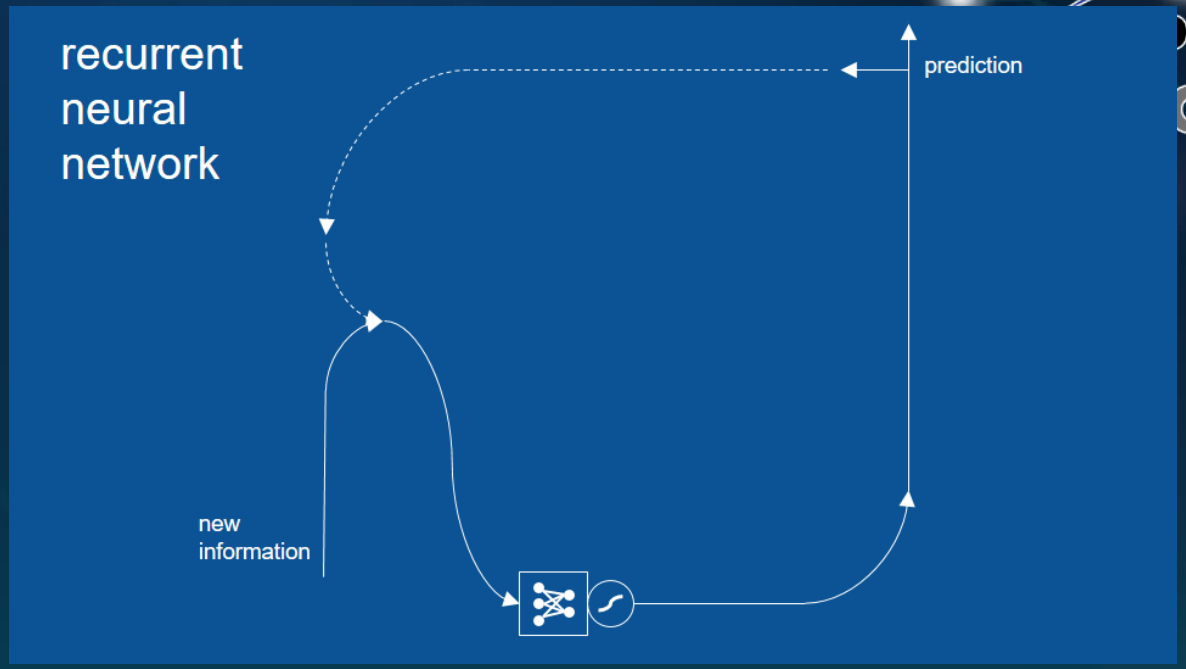
- 橫貫所有單元，帶著「**記憶**」的單元狀態(cell state)
- 右下圖中的黃色「**閘門**(邏輯函數)」，可用來決定上一層帶來的資訊，以及這一層新增加的資訊，有多少比例要被忘記，不帶入下一層。

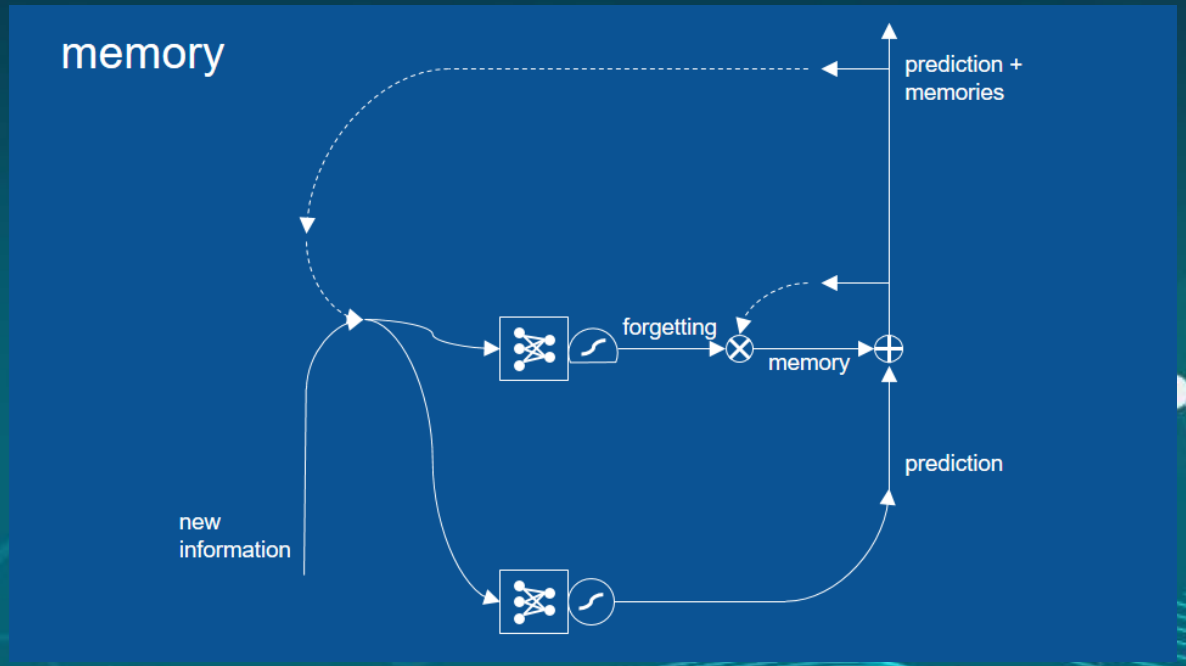




LSTM講解

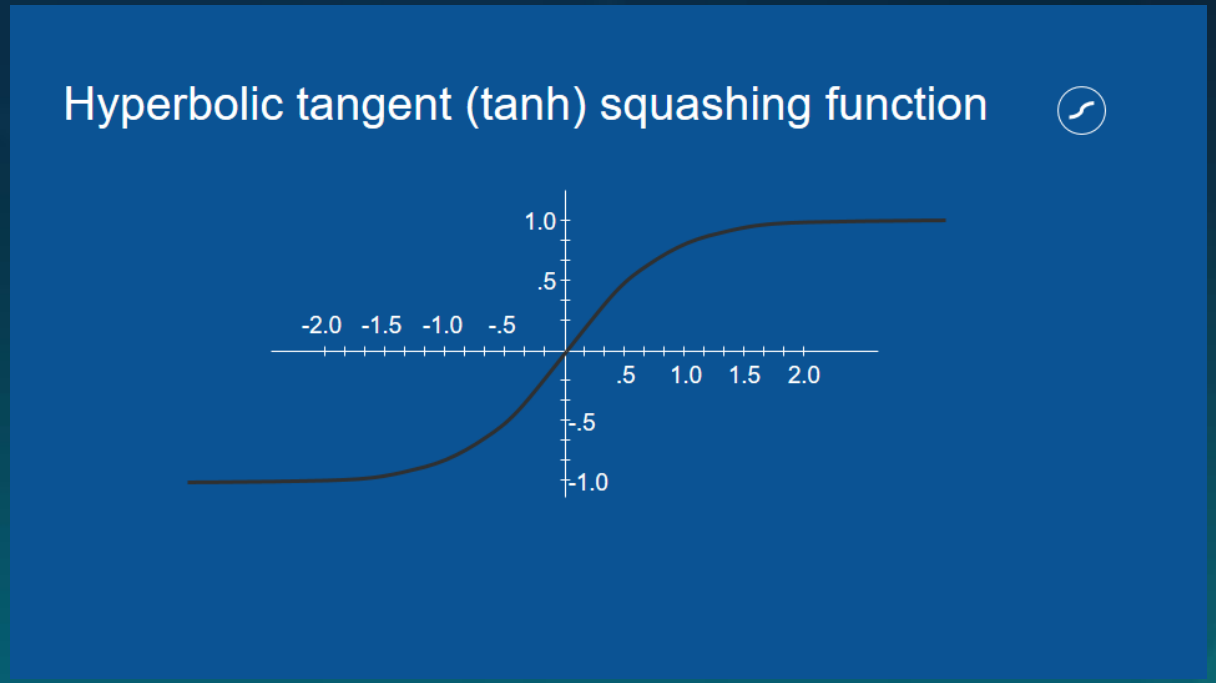
- 記憶／遺忘路徑
  - 右圖中，新增的關鍵是記憶／遺忘（memory and forgetting）路徑，用於幫助模型能記住幾個循環前發生的事情。為了解釋記憶部分的運作原理，需要先認識幾個新的符號。



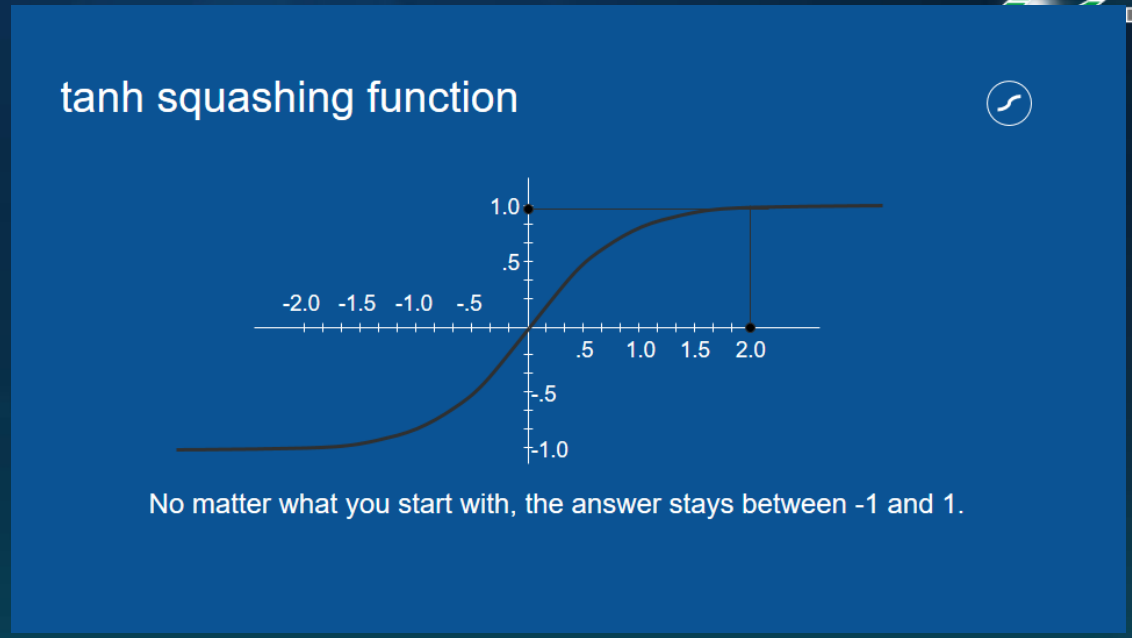


LSTM講解

- 擠壓函數（雙曲正切函數）
  - 這個波浪符號代表擠壓函數（squashing function，又譯作 S 函數），它可以幫助整個神經網路更好運作。



LSTM講解

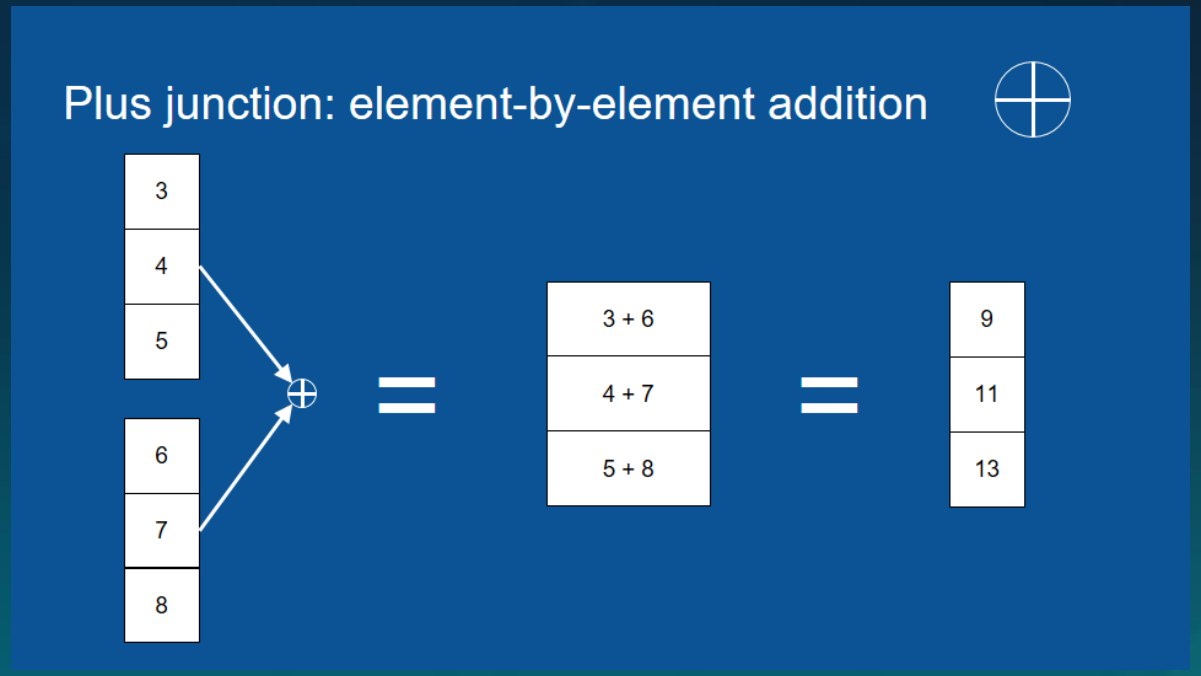


- 對於小的數值而言，原始數值和擠壓過構的數值通常很相近，但隨著數值增大，擠壓過後的數值會越來越接近 1。隨著數值愈趨負無窮大，擠壓過後的數值也會越來越接近 -1。不論如何，擠壓過後的數值都會介於 1 和 -1 之間。

擠壓函數的處理，對於神經網路這種重複運算相同數值的流程非常有用。比方說，如果有個選項每次都得到兩次投票，它的數值也會被乘以二，隨著流程重複，這個數字很容易被放大成天文數字。藉由確保數值介於 1 和 -1 之間，即使將數值相乘無數次，也不用擔心它會在循環中無限增大。這是一種負回饋（negative feedback）或衰減回饋（attenuating feedback）的例子。

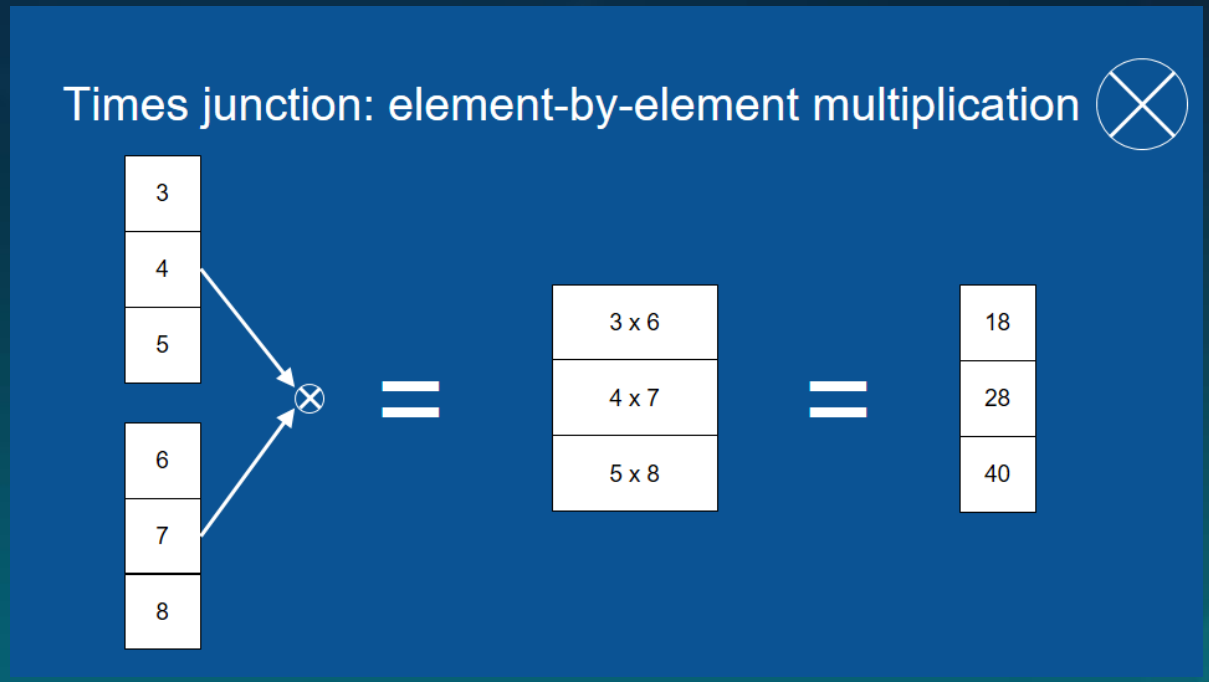
LSTM講解

- 逐元素相加、相乘和閘門
  - 首先，圈圈裡包含十字的符號是（矩陣）逐元素加法（element by element addition）。它的運作原理是將兩個相同長度的矩陣按同位置和順序的元素相加，所以可以將第一個向量中的第一個元素，和第二個向量中的第一個元素相加，得到輸出向量中的第一個元素。



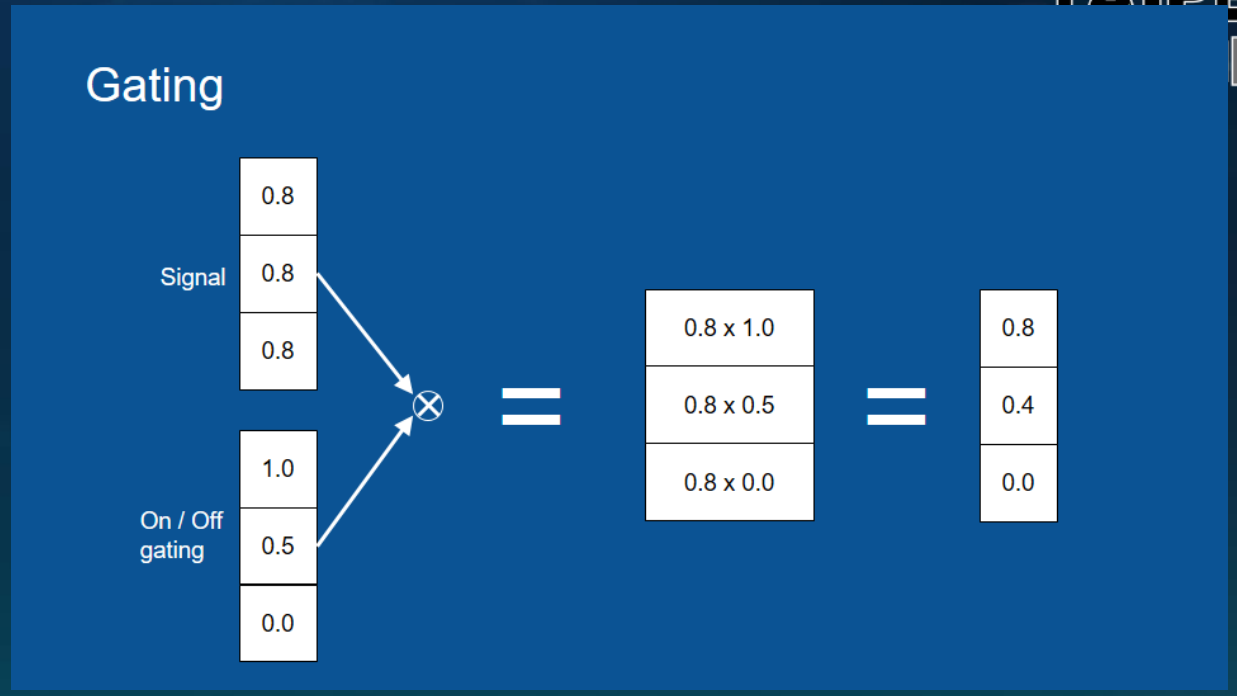
LSTM講解

- 於是在前頁圖中可以看到 3 加 6 等於 9。可以接著處理下一個元素：4 加 7 等於 11，最後所得到的向量會和原本長度一樣，不過其中每一個元素都是前兩個向量逐元素的和。
- 圈圈裡有個交叉的符號是（矩陣）逐元素乘法（element by element multiplication）。它和逐元素加法的運作原理類似，只是運算方法從加法換成了乘法。



LSTM講解

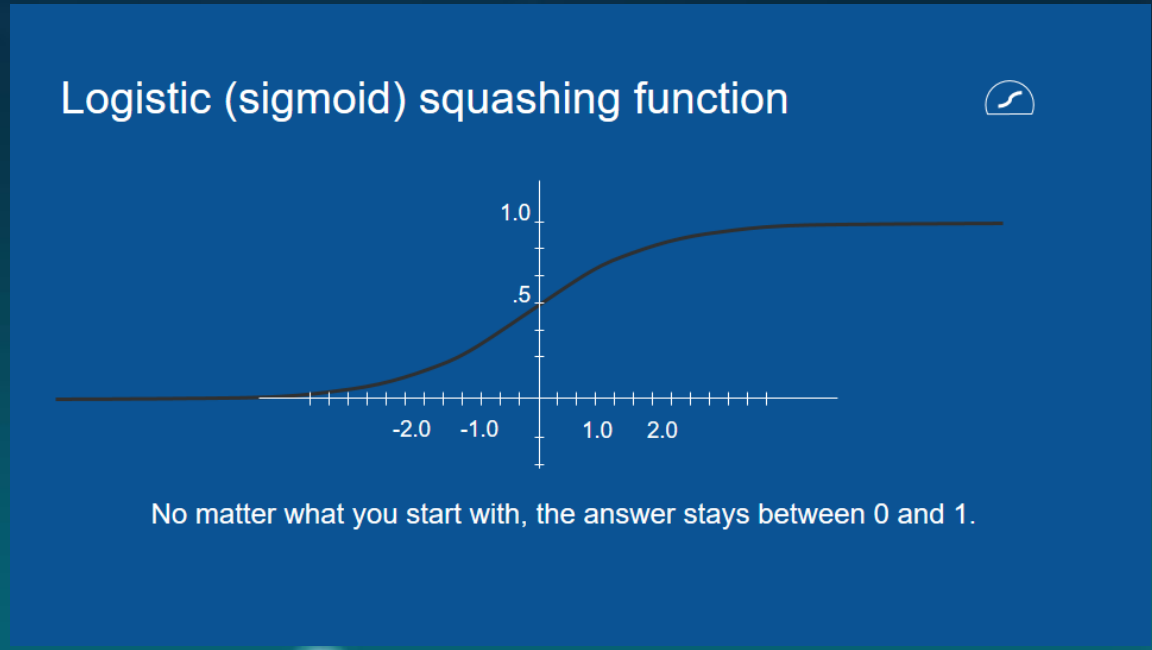
- 逐元素相乘可以幫助完成一些事情。可以想像有一組訊號，以及一組可以控制水量的水管。在這個例子裡，把初始訊號都當作 0.8。



在每個水管上都有一個龍頭，可以用來全開、全關或任意水量，讓訊號流通或堵塞。所以在這個例子裡，全開的龍頭有乘數 1，而全關的龍頭則有乘數 0。根據前面提到的逐元素乘法，可以在一開始將 0.8 乘上全開的 1，得到原本的訊號 0.8，也會在最後將 0.8 乘上 0，得到被遮蔽的訊號 0。中間的 0.8 則會乘上 0.5，得到一個比較小、衰減過後的訊號。這組閘門（gate）可以讓控制訊號的流通與否，非常有用。

LSTM講解

- 擠壓函數（邏輯函數）
  - 為了實現上述的閘門，需要一組介於 0 和 1 之間的數值，所以這裡又有另一種壓縮函數。這個函數的符號是一個帶有平底的圓形，它被稱作邏輯函數（logistic function）。邏輯函數和前面提到的雙曲正切函數（hyperbolic tangent function）很類似，除了前者的輸出值介於 0 和 1 之間，而非後者的 -1 和 1 之間。



LSTM基本原理

- 一層LSTM是由單個循環結構結構組成，而不是多個循環結構連接組成，即是說由輸入數據的維度和循環次數決定單個循環結構需要自我更新幾次，當前層LSTM的參數總個數只需計算一個循環單元就行，而不是計算多個連續單元的總個數。

- 關於這段描述，在實際操作的理解詳述請參考：Keras關於LSTM的units參數，還是不理解?

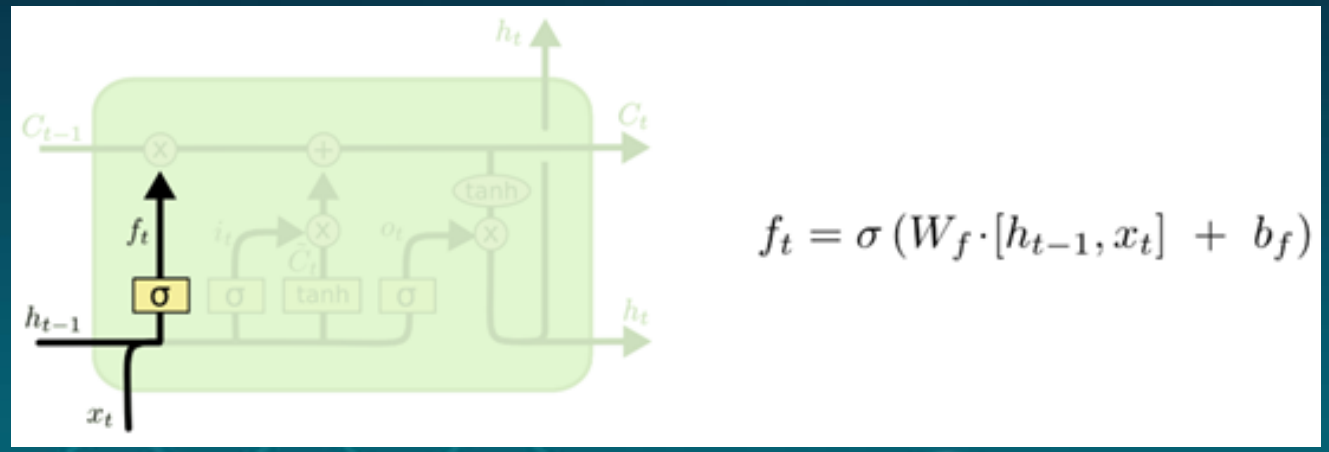
LSTM基本原理

- 下面將由一組圖來詳細結構LSTM cell的基本組成和實現原理。LSTM cell由**輸入門**、**遺忘門**、**輸出門**和**單元狀態**組成。

  - **輸入門**：決定當前時刻網路的輸入數據有多少需要保存到單元狀態。
  - **遺忘門**：決定上一時刻的單元狀態有多少需要保留到當前時刻。
  - **輸出門**：控制當前單元狀態有多少需要輸出到當前的輸出值。

LSTM基本原理

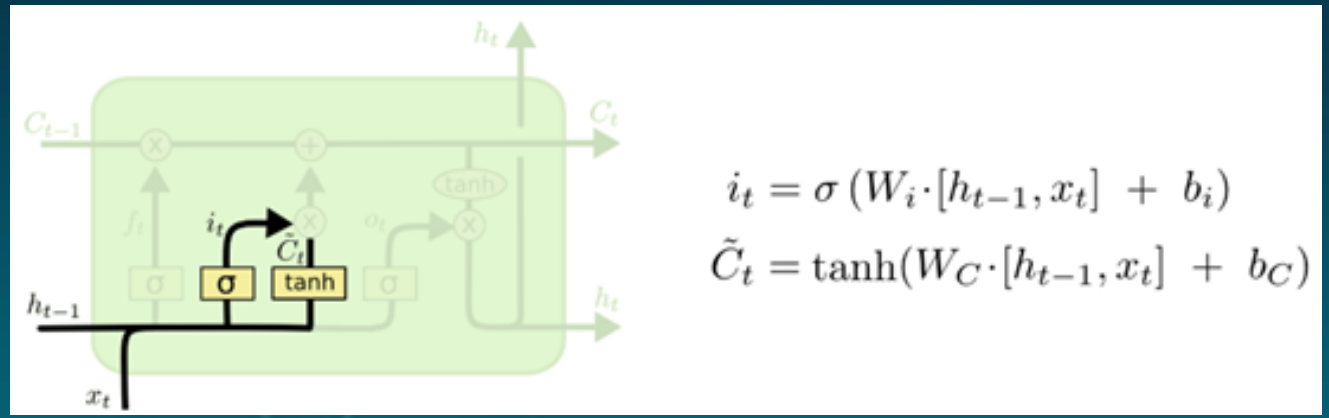
- 下圖展示了應用上一個時刻的輸出h_{t-1}和當前的數據輸入x_t，通過**遺忘門**得到f_t的過程。



- 原圖來源：Understanding LSTM Networks

LSTM基本原理

- 下圖展示了應用上一個時刻的輸出h_{t-1}和當前的數據輸入x_t，通過**輸入門**得到i_t，以及通過單元狀態得到當前時刻暫時狀態$\tilde{C}_t$的過程。

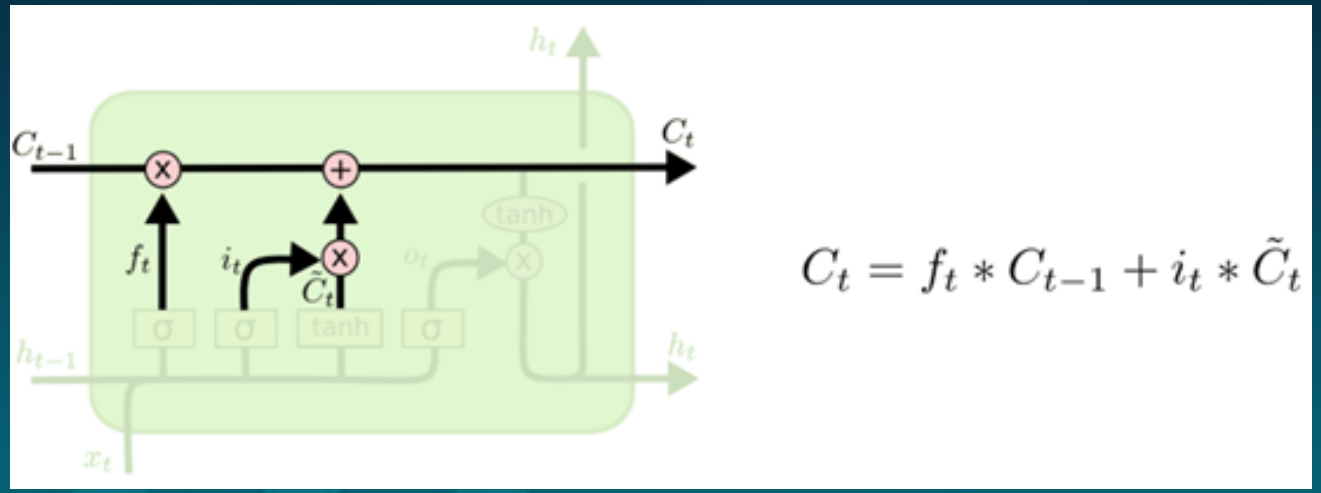


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM基本原理

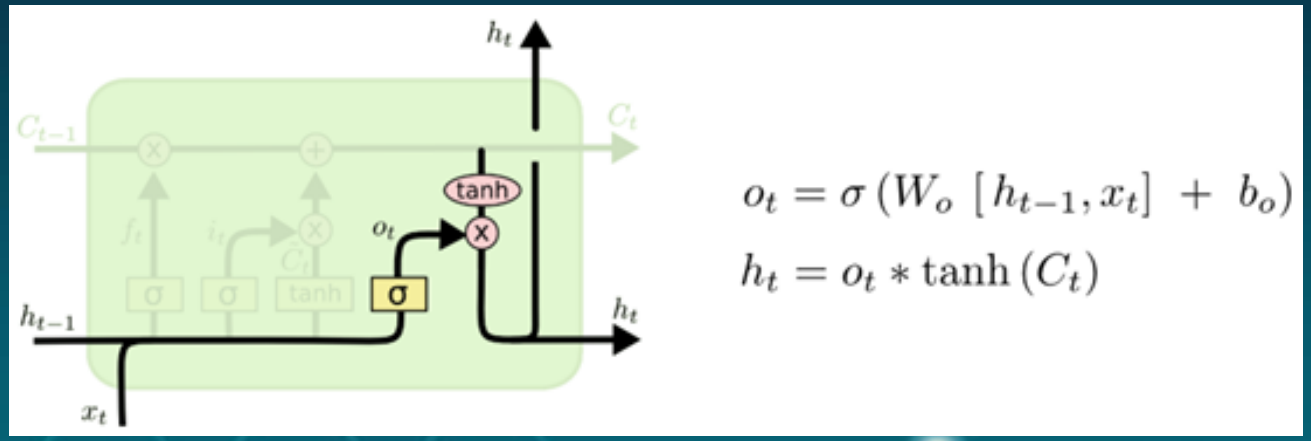
- 下圖展示了應用上一個cell結構的單元狀態C_{t-1}、**遺忘門**輸出f_t、**輸入門**輸出i_t，以及單元狀態的輸出$\tilde{C}_t$，得到當前cell的狀態C_t的過程。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM基本原理

- 下圖展示了應用上一個時刻的輸出h_{t-1}和當前的數據輸入x_t，通過**輸出門**得到O_t的過程，以及結合當前cell的單元狀態C_t和O_t得到最終的輸出h_t的過程。



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM程式碼範例

- LSTM訓練流程：
  1. 讀取資料、資料前處理
  2. 將資料分成train_set和test_set
  3. 建構LSTM神經網路模型、設定模型內一層層的神經網路
  4. 將資料送入網路中進行訓練
  5. 使用訓練完好的模型進行預測

Ref:https://zhuanlan.zhihu.com/p/81606029

LSTM程式碼範例

- 本程式碼為利用LSTM預測搭乘飛機的乘客客流量的範例(資料集取得可通過各國交通主管機關的統計資料，整理為.csv檔案格式進行讀取)。

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
```

引入所需的函式庫

```
# 讀取.csv檔案
data = pd.read_csv('international-airline-passengers.csv')
data['time'] = pd.to_datetime(data['time'])
data = data.set_index('time')
data.head()
```

讀取自己整理好的.csv格式資料集

data.head()可以展示所讀取的.csv格式資料集前5筆資料內容

time	passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

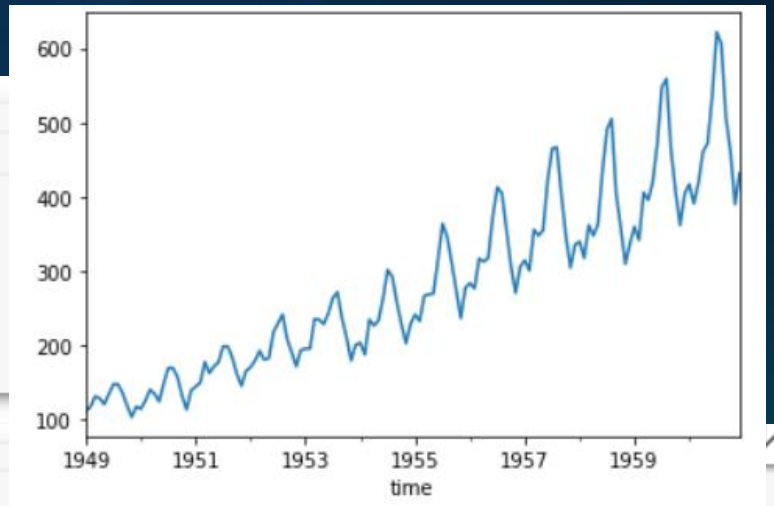
LSTM程式碼範例

```
#  劃出趨勢圖
def  get_picture(data  =  data):
        data['passengers'].plot()
        plt.figure(figsize=(100,50))
        plt.show()
```

定義函式：將所讀入的.csv檔案透過matplotlib進行可視化

以passenger畫出折線圖

設定趨勢圖寬度和長度(單位:inches)




```
#  轉化序列
def  processing(data=data,long=11):
        """
        依次轉化為11列
        """
        data['passengers']  =  data['passengers'].astype(float)
        sample  =  len(data)  -  long  +  1
        print('得到{}個樣本'.format(sample))
        data_sample  =  []
        for  i  in  range(sample):
                data_sample.append(data['passengers'][i:i+long])
        data_sample  =  np.array(data_sample)
        return  data_sample
```

定義函式：將所讀入的資料轉化為序列

將以passenger型態轉為float

將passenger資料每11筆append至新的變數data_sample中

LSTM程式碼範例

```
#  訓練LSTM網路
def  lstm(input_data=  None):
        scaler_x  =  MinMaxScaler()
        scaler_y  =  MinMaxScaler()
        x  =  input_data[:,:-1]
        y  =  input_data[:,-1]
        x  =  scaler_x.fit_transform(x)
        y  =  scaler_y.fit_transform(np.reshape(y,(len(y),1)))
        spilt  =  int(len(y)*0.8)
        x_train  =  x[:spilt]
        x_test  =  x[spilt:]
        y_train  =  y[:spilt]
        y_test  =  y[spilt:]
        x_train  =  np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
        x_test  =  np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

將數據進行歸一化

將input_data進行預處理

fit_transform()的作用是預訓練，找到轉換資料的規則，然後根據找到的規則轉換資料

將訓練集和測試集分割成8:2

將訓練集和測試集進行reshape

```
model = Sequential()
model.add(LSTM(50, input_shape=(x_train.shape[1], 1), return_sequences=True))
model.add(LSTM(100))
model.add(Dense(1, activation='linear'))
model.compile(loss='mse', optimizer='rmsprop')
print('Train...')
history = model.fit(x_train, y_train, batch_size=8, epochs=300, validation_split=0.1)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (12, 4.5))
#  畫出train_loss
ax1.set_title('train_loss')
ax1.set_ylabel('loss')
ax1.set_xlabel('Epoch')
ax1.plot( history.history["loss"])

predict = model.predict(x_test)
y_test = scaler_y.inverse_transform(np.reshape(y_test, (len(y_test), 1)))
predict = scaler_y.inverse_transform(predict)
#  print(type(predict))
#  print(predict)
#  劃出預測圖
ax2.set_title('prediction')
ax2.set_ylabel('passengers')
ax2.set_xlabel('time')
ax2.plot(predict, 'g:')
ax2.plot(y_test, 'r-')
fig.show()
```

初始化Sequential的模型

加入LSTM，50的意思為將輸入的維度映射成50個維度輸出

加入Dense並使用線性的激勵函數

以compile函數定義損失函數(loss)、最佳化函數(optimizer)

設定好batch_size、epochs訓練代數，

validation_split用於在沒有提供驗證集的時候，按一定比例從訓練集中取出一部分作為驗證集

畫出訓練過程中的train_loss折線圖

輸入資料以進行人流預測

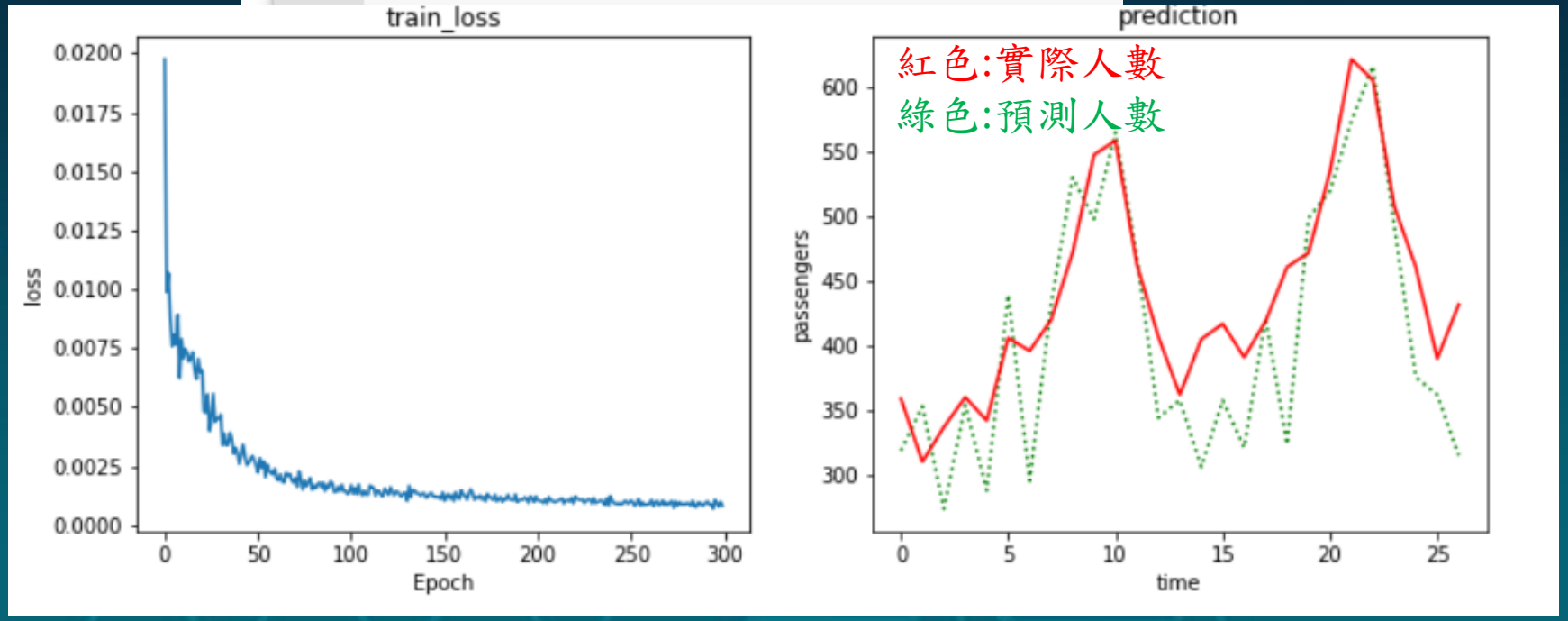
畫出預測結果

```
get_picture()
sam = processing()
lstm(sam)
```

畫出趨勢圖

轉化序列

進行LSTM訓練



- 其它RNN與LSTM的補充教材：
  - https://easyai.tech/ai-definition/rnn/?fbclid=IwAR2Wt9iF0vSbshRBLWr3gt1PnJ1G7knFMPNd8PJhJJ2swQZkwWLxkqXCRMw#jiazhi
  - https://brohrer.mcknote.com/zh-Hant/how_machine_learning_works/how_rnns_lstm_work.html?fbclid=IwAR00w4lFSUdTIgvVkc_WD7FfLL89gvPJxdn_22IP81_0Q5hKnBJBe7RAeIc
  - https://medium.com/ai-academy-taiwan/%E7%B0%A1%E5%96%AE%E6%89%8B%E5%88%BB-lstm-%E6%A8%A1%E5%9E%8B-69f06d9b0c94
- LSTM股票價格預測範例：
  - https://wenwender.wordpress.com/2019/10/18/%E5%AF%A6%E4%BD%9C%E9%80%8F%E9%81%8Elstm%E9%A0%90%E6%B8%AC%E8%82%A1%E7%A5%A8/ https://blog.csdn.net/wenshijie123/article/details/88764036